Ryan Aden George

CIA2-RL

Problem statement:

To implement a K-Armed Bandit-Based Recommendation system

How will this help?

The system should dynamically learn from user interactions to maximize cumulative rewards.

Algorithm design:

Choosing E-Greedy algo

CodeStructure:

k = 10

epsilon = 0.1

n\_iterations = 1000

Desired result:

1-Total Rewards

2-Average Reward

3-Estimated Q-values

4-Number of times each arm was selected

Sources:

-Medium

-ResearchPapers

-OpenAI(for errors)

# Action Space (Arms):

Each arm represents a different article available for recommendation at a given time.

The number of arms corresponds to the number of articles in the system (K).

import numpy as np

import random

class KArmBandit:

def \_\_init\_\_(self, k, epsilon=0.1, aligned\_weight=1.5):

self.k = k

self.epsilon = epsilon

self.aligned\_weight = aligned\_weight

self.q\_values = np.zeros(k)

self.arm\_counts = np.zeros(k)

def select\_arm(self):

if random.random() < self.epsilon:

return random.randint(0, self.k - 1)

else:

return np.argmax(self.q\_values)

def update(self, arm, reward, aligned):

if aligned:

reward \*= self.aligned\_weight

self.arm\_counts[arm] += 1

self.q\_values[arm] += (reward - self.q\_values[arm]) / self.arm\_counts[arm]

def run\_bandit(bandit, n\_iterations, reward\_function):

rewards = np.zeros(n\_iterations)

for i in range(n\_iterations):

arm = bandit.select\_arm()

aligned = random.choice([True, False])

reward = reward\_function(arm, aligned)

bandit.update(arm, reward, aligned)

rewards[i] = reward

return rewards

def reward\_function(arm, aligned):

base\_reward = np.random.normal(arm + 1, 1)

return base\_reward + (2 if aligned else 0)

k = 10

epsilon = 0.1

n\_iterations = 1000

bandit = KArmBandit(k=k, epsilon=epsilon)

rewards = run\_bandit(bandit, n\_iterations, reward\_function)

print(f"Total Reward: {np.sum(rewards)}")

print(f"Average Reward: {np.mean(rewards)}")

print(f"Estimated Q-values: {bandit.q\_values}")

print(f"Number of times each arm was selected: {bandit.arm\_counts}")